AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Original) A parser for parsing documents comprising a Document Order List Creator, a YPath Table Creator, and a ZPath Table Creator;
the arrangement of the Document Order List creator being such that it is adapted to produce a Document Order List (DOL) correlating a unique index number with an associated node, and the Document Order List creator being adapted to allocate each node in a document parsed its own unique index number in the DOL;
and wherein the arrangement of the YPath Table Creator is such that it is adapted to produce a YPath Table having a set of sequences of node names representative of all sequences of node names encountered in traversing from the root node to all possible nodes in the document parsed, and associated with each sequence of node names a corresponding group of index numbers corresponding to the index numbers in the DOL for those nodes in the DOL for which the associated sequence of node names is true for a traversed pathway from the root node to the specified node, each node name sequence encounterable in parsing from the root node to any other node in the document appearing only once in the YPath Table; and wherein the arrangement of the ZPath Table Creator is adapted to create a ZPath Table having a set of sequences of ordinals representative of ordinals of those nodes encounterable in traversing from the root node to all possible nodes in the document, the ordinal of a node being the integer value position of the node amongst those nodes of the same name which share the same parent node, and associated with each set of sequences of ordinals, a corresponding group of index numbers corresponding to the index numbers in the DOL for those nodes in the DOL for which the associated sequence of ordinals for nodes is true for a traversed pathway from the root node to the specified node, each sequence of ordinals encounterable in parsing from the root node to any other node appearing only once in the

ZPath Table.

2. (Original) A parser according to claim 1 wherein the parser is adapted to create the DOL as an ordered list of nodes encountered in the DOL when the document is parsed.

3. (Original) A parser according to claim 2 wherein the DOL Creator is arranged to produce the DOL depth first, in the order that nodes appear in the document.

4. (Currently Amended) A parser according to ~~any preceding claim~~ claim 1 wherein (i) the YPath creator is arranged to produce the YPath Table depth first; and/or (ii) the ZPath Creator is arranged to produce the ZPath Table depth first.

5. (Currently Amended) A parser according to ~~any preceding claim~~ claim 1 adapted to associate in the DOL a type of node category with nodes.

6. (Original) A parser according to claim 5 wherein the type associated with a node includes: element nodes, attribute nodes, and text nodes.

7. (Currently Amended) A parser according to ~~any preceding claim~~ claim 1 wherein the DOL Creator is adapted to create a DOL having a column for index numbers and at least one, or any combinations of columns for:
(i) node type
(ii) node name
(iii) node value.

8. (Currently Amended) A parser according to ~~any preceding claim~~ claim 1 adapted to parse XML documents.

9. (Original) A data structure representative of a document comprising a Document Order List (DOL), a Node Name Sequence List, and an Ordinal Sequence List;
the DOL having a correlation of each node in the document with a unique index number;
the Node Name Sequence List having a correlation of (i) each possible sequence of node names encountered in traversing the document from the root node to all nodes with (ii) the index numbers in the DOL associated with nodes for which each particular node name sequence is true;
the Ordinal Sequence List having a correlation of (i) each node name ordinal sequence that it is possible to have in traversing the document from the root node to all nodes with (ii) the index numbers in the DOL associated with the nodes in the DOL for which each particular node name ordinal sequence is true.

10. (Original) A data structure according to claim 9 which represents an XML document, or other document represented as a tree of connected nodes.

11. (Original) A data structure according to claim 9 wherein the DOL comprises an index number column associating unique index numbers with each node in the document; and at least one of:-
(i) a column associating node names with index numbers, for at least some nodes;
(ii) a column associating node type with index numbers, for at least some nodes;
(iii) node value with index numbers, for at least some nodes.

12. (Original) A data structure according to claim 11 having all three of (i), (ii) and (iii).

13. (Original) A data structure according to claim 11 wherein node types associable with a node include one, two or three of: element, attribute and text.

14. (Original) A method of querying a data structure representative of a document, the data

structure being in accordance with claim 9 comprising the steps of:-

querying the Node Name Sequence List and/or the Ordinal Sequence List for a target node or nodes to identify index numbers associated with the target node and returning the node or nodes associated with identified index numbers in the DOL.

15. (Original) A method according to claim 14 further comprising returning all values in the DOL associated with all index numbers identified by querying:

(i) the Node Sequence List alone; or

(ii) the Ordinal Sequence List alone; or

(iv) the common, intersection, index numbers present in querying both the Node Name Sequence List and the Ordinal Sequence List.

16. (Original) A method according to claim 15 wherein the query of the Node Sequence List produces a first sequence of index numbers and the query of the Ordinal Sequence List produces a second sequence of index numbers and wherein the common index number or numbers present in both the first and second sequences of index numbers is identified by partitioning each of the two lists into a lower index number range and a higher index number range divided by dividing points respective to each sequence, and comparing index numbers from the lower index number range of the first sequence with index numbers from the lower index number range of the second sequence, and comparing index numbers from the higher index number range of the first sequence with index numbers from the higher index number range of the second sequence.

17. (Original) A method according to claim 15 comprising dividing the first and second index number sequences by splitting them into upper and lower ranges and pairing first and second upper and lower sources respectively, to create pairs of index number sequences from the YPath and ZPath returns, and further dividing said pairs to create subsequent generation pairs of YPath and ZPath returns until a match is found between index numbers of said pairs, or subsequent generation

pairs, of YPath and ZPath index number sequences.

18. (Original) A method according to claim 15 comprising comparing both the lowest and highest index numbers in the YPath return sequence of index numbers with both the highest and lowest index numbers of a ZPath return sequence of index numbers, and optionally also comparing the mid point index number in the YPath return with the mid point index number in the ZPath return.

19. (Currently Amended) A method according to ~~any one of~~ claim 15 comprising comparing the lowest and/or highest index number of the one of (i) the Y Path return sequence of index numbers, or (ii) the ZPath return sequence of index numbers, with a mid point index number from the other of (i) or (ii).

20. (Original) A method according to claim 15 wherein a pair of index number sequences representative of a YPath Query return and a ZPath Query return have index numbers in each of the pair of index number sequences removed from consideration for being the intersection, the removed index numbers comprising:

(i) those index numbers from one sequence of the pair that are lower than the lowest index number in the other sequence of the pair, and vice-versa, those index numbers from said other sequence of the pair that are lower than the lowest index number in said one sequence of the pair; and
(ii) those index numbers from one sequence of the pair that are higher than the highest index number of the sequence of the pair, and vice versa, those index numbers from said other sequence of the pair that are higher than the highest index number from said one sequence of the pair.

21. (Currently Amended) A method according to ~~any one of~~ claim 15 wherein the following checks are performed to determine if a matching index number can be found in the YPath return sequence of index numbers and the ZPath return sequence of index numbers, said checks comprising at least one,

some, or all of:

(i) establishing whether the minimum index number of the first sequence equals the minimum index number of the second sequence;

(ii) establishing whether the maximum index number of the first sequence equals the maximum index number of the second sequence;

(iii) establishing whether the minimum index number of the first sequence equals the maximum index number of the second sequence;

(iv) establishing whether the minimum index number of the second sequence equals the maximum index number of the first sequence;

(v) establishing whether the mid index number of the first sequence equals the mid index number of the second sequence.

22. (Original) A method according to claim 15 wherein checks are performed to determine if a matching index number can be found in the YPath return sequence of index numbers and the ZPath return sequence of numbers, said checks comprising at least one, some, or all of:

(i) establishing whether the minimum index number of the first sequence equals the mid index number of the second sequence;

(ii) establishing whether the maximum index number of the first sequence equals the mid index number of the second sequence;

(iii) establishing whether the mid index number of the first sequence equals the minimum index number of the second sequence;

(iv) establishing whether the mid index number of the first sequence equals the maximum index number of the second sequence.

23. (Original) A method according to claim 21 wherein the mid index number of a sequence is consistently taken as the number below, or consistently taken as the number above, the mid point if there is an even number of index numbers in the sequence.

24. (Original) A computer memory holding a YPath Table, an XPath Table and a Document Order List derived from and representative of a document to be queried, the Document Order List comprising a list of nodes in the document and an index number uniquely associated with each node; and wherein

the YPath Table comprises a set of sequences of node names traversed in navigating from a root node to each specific node in the document, and associated with each particular sequence of node names a corresponding list of index numbers representative of the nodes in the document for which that particular node name sequence is true; and

the ZPath Table comprises a set of sequences of ordinals of node names encountered when traversing the document from the root node to any node in the document, and associated with each ordinal sequence the index numbers equivalent to the nodes for which that ordinal sequence for nodes traversed is true.

25. (Original) A computer according to claim 24 wherein the index numbers associated with each node name sequence in the YPath Table and/or the ZPath Table are ordered in the sequence in which they are encountered when the document is parsed.

26. (Original) A method of making an XPath query comprising resolving the query into a YPath query and a ZPath query, and querying a YPath Node Name Sequence Table with the YPath query and a ZPath Node Sequence Table with the ZPath query, where YPath is a node name sequence of an XPath sequence but with no ordinals, and where ZPath is an ordinal sequence for a node name sequence of an XPath query, but with no node names.

27. (Original) A method of holding data representative of a document in a computer readable memory comprising storing in the memory a data structure representative of a document comprising a Document Order List, a YPath Table, and a ZPath Table; the Document Order List comprising a

correlation between (i) each node in the document being represented and (ii) a unique index number;
a YPath Table comprising a correlation between (i) each possible node name sequence traversable in the document to reach a node in the document and (ii) the index numbers for nodes which are locatable using that node name sequence;
a ZPath Table comprising a correlation between (i) each possible ordinal sequence for nodes traversed to reach a node in the document and (ii) the index numbers for the nodes which are locatable using that ordinal sequence.

28. (Original) A method according to claim 27 wherein the document is parsed in a depth first ordering system.

29. (Original) A method according to claim 27 wherein the document is parsed in a breadth first ordering system.

30. (Original) Computer program product for querying a data structure in accordance with claim 9, the software comprising:
a YPath query engine adapted in use when operating on a computer processor, to make a YPath query of the YPath Table for a node or nodes locatable with a specified node name sequence and to return index numbers for nodes satisfying the query;
a ZPath query engine adapted in use when operating on a computer processor, to make a ZPath query of the ZPath Table for a node or nodes locatable with a specified node ordinal sequence and to return index numbers for nodes satisfying the query;
and an intersection locator engine adapted in use, when operating on a computer processor, to identify a node or nodes common to the returns of the YPath and ZPath queries and to return said common nodes.

31. (Original) A computer having a computer processor, a program memory, and access to a data

structure memory, the processor being adapted to run a YPath and ZPath query of a data structure held on said data structure memory using YPath and ZPath query engine and intersection determining software held in said program memory; and the computer processor being adapted to run document parsing software held on said program memory adapted to parse a document and create and store a Document Order List, a YPath Table and a ZPath Table representative of said document in said data structure memory;

said Document Order List comprising a concordance between (i) each node in the document and (ii) an associated unique index number;

said YPath Table comprising a concordance between (i) each possible node name sequence traversable in the document when parsing it and (ii) index numbers corresponding to nodes locatable using each node name sequence;

said ZPath Table comprising a concordance between (i) ordinal numbers of each possible node path in said document to each node in said document and (ii) index numbers representative of nodes locatable using each ordinal number sequence;

said YPath query engine being adapted to return index numbers corresponding to nodes for which a specified queried node name sequence is true;

said ZPath query engine being adapted to return index numbers corresponding to nodes for which a specified queried ordinal number sequence is true;

said intersection determining software being adapted to identify intersection index numbers present in both the YPath and ZPath query engine returns and to output nodes determined by said intersection index numbers;

said processor being capable of parsing a document to create a document order list, YPath Table, and ZPath Table corresponding to said document and also to query said Document Order List, YPath Table and ZPath Table to retrieve a response to its query.

32. (Original) A computer having a computer processor, a program memory, and a data structure memory, the processor being adapted to run a YPath and ZPath query of a data structure

representative of an XML document held on said data structure memory using YPath and ZPath query engine and intersection determining software held in said program memory; and the computer processor being adapted to run document parsing software held on said program memory adapted to parse an XML document and create and store a Document Order List , a YPath Table and a ZPath Table derived from said XML document in said data structure memory;
said Document Order List comprising a concordance between (i) each node in the XML document and (ii) an associated unique index number;
said YPath Table comprising a concordance between (i) each possible node name sequence traversable in the XML document when parsing it and (ii) index numbers corresponding to nodes locatable using each node name sequence;
said ZPath Table comprising a concordance between (i) ordinal numbers of each possible node path in said XML document to each node in said XML document and (ii) index numbers representative of nodes locatable using each ordinal number sequence;
said YPath query engine being adapted to return index numbers corresponding to nodes for which a specified queried node name sequence is true;
said ZPath query engine being adapted to return index numbers corresponding to nodes for which a specified queried ordinal number sequence is true;
said intersection determining software being adapted to identify intersection index numbers present in both the YPath and ZPath query engine returns and to output nodes determined by said intersection index numbers;
said processor being capable of parsing a document to create a document order list, YPath Table, and ZPath Table corresponding to said document and also to query said Document Order List, YPath Table and ZPath Table to retrieve a response to its query.

33. (Original) A computer having a computer processing means, YPath query means, ZPath query means, intersection determining means, document parsing means and data structure means, the processing means being adapted to operate the YPath and ZPath query means to query the data

structure means and to operate the intersection determining means; and the computer processing means also being adapted to operate the document parsing means to parse a document and create and store a Document Order List, a YPath Table and a ZPath Table representative of said document in said data structure means;

said Document Order List comprising a concordance between (i) each node in the document and (ii) an associated unique index number;

said YPath Table comprising a concordance between (i) each possible node name sequence traversable in the document when parsing it and (ii) index numbers corresponding to nodes locatable using each node name sequence;

said ZPath Table comprising a concordance between (i) ordinal numbers of each possible node path in said document to each node in said document and (ii) index numbers representative of nodes locatable using each ordinal number sequence;

said YPath query means being adapted to return index numbers corresponding to nodes for which a specified queried node name sequence is true;

said ZPath query means being adapted to return index numbers corresponding to nodes for which a specified queried ordinal number sequence is true;

said intersection determining means being adapted to identify intersection index numbers present in both the YPath and ZPath query means returns and to output nodes determined by said intersection index numbers;

said processing means being capable of parsing a document to create a document order list, YPath Table, and ZPath Table corresponding to said document and also to query said Document Order List, YPath Table and ZPath Table to retrieve a response to its query.

34. (Original) A document parser comprising:

a correlator adapted to allocate to each node in a document a unique index number;

a YPath table generator adapted to associate, to each of a plurality of unique sequences of node names which are encountered in traversing to a given node from a root node, index numbers for

which an associated sequence of node names is true; and

a Zpath table generator adapted to associate, to each of a plurality of unique sequences of ordinal numbers (of relative node position amongst nodes of the same name and sharing a common parent node) which are encountered in traversing to a given node from the root node, index numbers for which an associated sequence of ordinal numbers is true.

35. (Original) A document parser according to claim 34 wherein the correlator is adapted to generate a list of nodes encountered when the document is parsed.

36. (Original) A document parser according to claim 36 wherein the correlator is adapted to list the nodes in the order they appear in the document.

37. (Original) A document parser according to claim 34 wherein the YPath and/or ZPath table generators are adapted to list sequences in the order they appear in the document.

38. (Original) A computer adapted to run the document parser of claim 34, and comprising means for establishing an index number which is true for both a sequence of node names and a sequence of ordinal numbers.